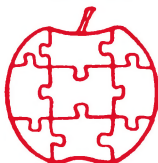


# Apple

\$1.80



# Assembly Line

Volume 4 -- Issue 5

February, 1984

## In This Issue...

Listing Buried Messages . . . . .	2
Peeking at the Catalog . . . . .	6
Fast Scroll for //e 80-Column. . . . .	8
DOS 3.3 Checksummer Debate Update. . . . .	10
So That's a Macintosh! . . . . .	11
Reminder about Wrap-Around Addressing. . . . .	12
Delays, Delays, Delays . . . . .	14
Annotated 68000 Bibliography . . . . .	19
Table of //e Soft Switches . . . . .	20
Text Area Erase Routine. . . . .	22
Amazing "quikLoader" Card. . . . .	27
Macro to Generate Quotient/Remainder Table for Hi-Res. . . . .	28

## Yes, ProDOS is Now Being Shipped

We bought an Apple //e last weekend, and it came with system disks for both DOS 3.3 and ProDOS. There was no DOS Reference Manual, although a little of DOS is mentioned in the Owner's manual. There is a very nice ProDOS User's Manual, 150 pages of text and photos and drawings. The dealer says he still has no word on ProDOS as a separate product.

## I Can't Believe He Typed The Whole Thing!

One of our readers took a few evenings and typed in the source code of the whole CX ROM from the Apple //e Reference Manual Addendum. This is the code from \$C100 through \$CFFC, which is listed on pages 23-49. He added some of his own comments to the source, which more fully explain what is going on in there. The source for the F8 ROM is on the disk too, but without many comments (pages 3-18 of the addendum). Naturally, the source files are in the format of the S-C Macro Assembler.

We think having the source of these ROMs on disk could enhance the //e in two ways: you can make a larger size copy of the listings, so they can be read in normal room light; and you can experiment with improvements to the code. If you have a PROM burner that will burn 2764s, I think you can even replace the chips. If you'd like a copy, send us \$15: we'll mail the disk to you, and pass along a percentage to the energetic typist.

**Listing Buried Messages.....Bob Sander-Cederlof**

Do you like treasure hunts? Dis-assembling, analyzing, understanding, and modifying programs written in assembly language, with nothing to go by but the program in memory and maybe a user's manual ... to me it is a treasure hunt.

Last week I desperately need to make full use of a Novation Cat II Modem. "Full use" of almost any peripheral device implies the use of assembly language. Even though Novation includes a very nice manual for the purpose, it did not answer half my questions.

Novation also includes a disk with a program called Com-Ware II. This program is assembly language, and takes 74 sectors on the disk. Somewhere, hidden in a small, dark corner, guarded by gnomes, surrounded by wild beasts, lay the answers to all my questions.

I started by BLOADing the file. Then "CALL -151" to get into the monitor, and typed "AA60.AA73". The first two bytes displayed the length of the file, and the last two bytes are the starting address. I learned it loaded at \$900, and was \$4825 bytes long.

I started using the monitor L command to scan through the program, and discovered that the programmer had placed all the screen messages "in line". That is, rather than putting all the screen text at the end of the whole program, or in the middle, or wherever, he coded the ASCII strings right in place. Each message was preceded by "JSR \$3866", and ended with a \$00 byte. The subroutine at \$3866 retrieved the return address from the stack, used it to address the message text while printing it out, and then placed a new return address on the stack to continue execution right after the \$00 byte.

This makes it difficult to use a program like Rak-Ware's wonderful DISASM, because you have to tell the boundaries of all non-executable code. And there seemed to be LOTS of messages.

On the other hand, it also makes it easier to follow the flow of the program. The buried messages are almost like living comments, telling me exactly what is going on in every section of code.

I decided to get my Apple to help. I wrote a "quick and dirty" program to scan through the whole image from \$900 through \$5125, looking for every occurrence of "JSR \$3866". I printed out the address of the next byte, which is the first byte of message text. Then I searched for the terminating \$00 byte, and printed out its address. Then I went back and printed out the message text.

After several tries, I even made my quick and dirty program nice and clean. I printed all the messages out, nicely formatted for easy visual scanning. I set my printer on 8 lines/inch and 12 chars/inch to save paper, and let 'er rip.

S-C Macro Assembler Version 1.0.....\$80.00  
 S-C Macro Assembler Version 1.1 Update.....\$12.50  
 Full Screen Editor for S-C Macro Assembler..... \$49.00  
 Includes complete source code.  
 S-C Cross Reference Utility.....\$20.00  
 S-C Cross Reference Utility with Complete Source Code.....\$50.00  
 DISASM Dis-Assembler (RAK-Ware).....\$30.00  
 Quick-Trace (Anthro-Digital).....(reg. \$50.00) \$45.00  
 The Visible Computer: 6502 (Software Masters).....(reg. \$50.00) \$45.00

S-C Word Processor (the one we use!).....\$50.00  
 With fully commented source code.  
 Applesoft Source Code on Disk.....\$50.00  
 Very heavily commented. Requires Applesoft and S-C Assembler.  
 ES-CAPE: Extended S-C Applesoft Program Editor.....\$60.00

AAL Quarterly Disks.....each \$15.00  
 Each disk contains all the source code from three issues of "Apple  
 Assembly Line", to save you lots of typing and testing time.  
 QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981  
 QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981 QD#6: Jan-Mar 1982  
 QD#7: Apr-Jun 1982 QD#8: Jul-Sep 1982 QD#9: Oct-Dec 1982  
 QD#10: Jan-Mar 1983 QD#11: Apr-Jun 1983 QD#12: Jul-Sep 1983  
 QD#13: Oct-Dec 1983

Double Precision Floating Point for Applesoft.....\$50.00  
 Provides 21-digit precision for Applesoft programs.  
 Includes sample Applesoft subroutines for standard math functions.  
 Amper-Magic (Anthro-Digital).....(reg. \$75.00) \$67.50  
 Amper-Magic Volume 2 (Anthro-Digital).....(reg. \$35.00) \$30.00  
 Routine Machine (Southwestern Data Systems).....(reg. \$64.95) \$60.00  
 FLASH! Integer BASIC Compiler (Laumer Research).....\$79.00  
 Fontrix (Data Transforms).....\$75.00  
 Aztec C Compiler System (Manx Software).....(reg. \$199.00) \$180.00  
 IACalc Spreadsheet Program.....(reg. \$84.95) \$75.00  
 The one we use every day. It's better than Visicalc!

Blank Diskettes.....package of 20 for \$45.00  
 (Premium quality, single-sided, double density, with hub rings)  
 Vinyl disk pages, 6"x8.5", hold one disk each.....10 for \$6.00  
 Diskette Mailing Protectors.....10-99: 40 cents each  
 100 or more: 25 cents each  
 Cardboard folders designed to fit into 6"x9" Envelopes.  
 Envelopes for Diskette Mailers..... 5 cents each  
 ZIF Game Socket Extender.....\$20.00

Buffered Grappler+ Interface and 16K Buffer.....(\$239.00) \$200.00  
 quikLoader EPROM Card.....(\$179.50) \$179.00

Books, Books, Books.....compare our discount prices!  
 "The Apple II Circuit Description", Gayler.....(\$22.95) \$21.00  
 "Understanding the Apple II", Sather.....(\$22.95) \$21.00  
 "Enhancing Your Apple II, vol. 1", Lancaster.....(\$17.95) \$17.00  
 "Incredible Secret Money Machine", Lancaster.....(\$7.95) \$7.50  
 "Beneath Apple DOS", Worth & Lechner.....(\$19.95) \$18.00  
 "Bag of Tricks", Worth & Lechner, with diskette.....(\$39.95) \$36.00  
 "Assembly Lines: The Book", Roger Wagner.....(\$19.95) \$18.00  
 "What's Where in the Apple", Second Edition.....(\$24.95) \$23.00  
 "What's Where Guide" (updates first edition).....(\$9.95) \$9.00  
 "6502 Assembly Language Programming", Leventhal.....(\$18.95) \$18.00  
 "6502 Subroutines", Leventhal.....(\$17.95) \$17.00

Add \$1.50 per book for US shipping. Foreign orders add postage needed.

\*\*\* S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 \*\*\*  
 \*\*\* (214) 324-2050 \*\*\*  
 \*\*\* We accept Master Card, VISA and American Express \*\*\*

Six whole pages! I think a third of Com-Ware is taken up by messages!

Here is a sample of the printout. Notice that I printed control characters, including <RETURN>, as "^M" followed by the printing form of the character. Thus "^M" means <RETURN>.

1481...14CC ^M^M<0> SET HIGH BIT^M<1> CLEAR HIGH BIT^M<2> LEAVE HIGH BIT ALONE^MCHOICE(0-2):

14EF...152E ^M^M<0> SEND LF AFTER CR^M<1> DO NOT SEND LF AFTER CR^MCHOICE(0-1):

I believe a lot of programs of interest use a similar technique for message printing, and slight adaptation of my MESSAGE SEARCH program could help YOU find some buried treasure!

```

1000 *SAVE S.MESSAGE SEARCH
1010 *-----
1020 *   FIND ALL MESSAGES IN COM-WARE II VERSION 5.0-3
1030 *
1040 *   ALL MESSAGES ARE PRECEDED BY "JSR $3866"
1050 *   AND END WITH A $00 BYTE:
1060 *
1070 *   20 66 38 <MSG> 00
1080 *-----
0000- 1090 MSG.PNTR .EQ $00,01
0002- 1100 END.PNTR .EQ $02,03
1110 *-----
F941- 1120 PRINTAX .EQ $F941
FD4D- 1130 COUT .EQ $FD4D
FD8E- 1140 CROUT .EQ $FD8E
1150 *-----
C000- 1160 KEYBOARD .EQ $C000
C010- 1170 STROBE .EQ $C010
1180 *-----
0800- A9 00 1190 FIND LDA #$900 COMWARE WAS BLOADED AT $900
0802- 85 00 1200 STA MSG.PNTR
0804- A9 09 1210 LDA /$900
0806- 85 01 1220 STA MSG.PNTR+1
0808- A5 00 1230 .1 LDA MSG.PNTR
080A- C9 25 1240 CMP #$5125 COMWARE ENDS AT $5125
080C- A5 01 1250 LDA MSG.PNTR+1
080E- E9 51 1260 SBC /$5125
0810- 90 01 1270 BCC .2 ...NOT AT END YET
0812- 60 1280 RTS ...FINISHED
1290 *---SEARCH FOR A $20 BYTE-----
0813- A0 00 1300 .2 LDY #0
0815- B1 00 1310 LDA (MSG.PNTR),Y
0817- C9 20 1320 CMP #$20
0819- F0 05 1330 BEQ .4 FOUND $20
081B- 20 A7 08 1340 .3 JSR INC
081E- D0 E8 1350 BNE .1 ...ALWAYS
1360 *---CHECK FOR $66, $38 AFTER $20---
0820- C8 1370 .4 INY
0821- B1 00 1380 LDA (MSG.PNTR),Y
0823- C9 66 1390 CMP #$66
0825- D0 F4 1400 BNE .3
0827- C8 1410 INY
0828- B1 00 1420 LDA (MSG.PNTR),Y
082A- C9 38 1430 CMP #$38
082C- D0 ED 1440 BNE .3
1450 *---FOUND A MESSAGE!-----
082E- A2 0A 1460 LDX #10
0830- 20 CA 08 1470 JSR MARGIN
0833- 20 AE 08 1480 JSR PAUSE
0836- 20 A7 08 1490 JSR INC SKIP OVER THE $20, $66, $38
0839- 20 A7 08 1500 JSR INC
083C- 20 A7 08 1510 JSR INC

```

```

083F- A5 01 1520 LDA MSG.PNTR+1 PRINT STARTING ADDRESS
0841- 85 03 1530 STA END.PNTR+1
0843- A6 00 1540 LDX MSG.PNTR
0845- 86 02 1550 STX END.PNTR
0847- 20 41 F9 1560 JSR PRINTAX
1570 *---SEARCH FOR END OF STRING-----
084A- A0 00 1580 LDY #0
084C- B1 02 1590 .5 LDA (END.PNTR),Y
084E- F0 08 1600 BEQ .6 FOUND END
0850- E6 02 1610 INC END.PNTR
0852- D0 F8 1620 BNE .5
0854- E6 03 1630 INC END.PNTR+1
0856- D0 F4 1640 BNE .5
1650 *---FOUND END OF STRING-----
0858- A9 AE 1660 .6 LDA #". " PRINT "... "
085A- 20 ED FD 1670 JSR COUT
085D- 20 ED FD 1680 JSR COUT PRINT THE END ADDRESS
0860- 20 ED FD 1690 JSR COUT
0863- A5 03 1700 LDA END.PNTR+1
0865- A6 02 1710 LDX END.PNTR
0867- 20 41 F9 1720 JSR PRINTAX
086A- A9 A0 1730 LDA #A0 PRINT " "
086C- 20 ED FD 1740 JSR COUT
086F- 20 ED FD 1750 JSR COUT
0872- 20 ED FD 1760 JSR COUT
1770 *---PRINT OUT THE STRING-----
0875- A0 00 1780 LDY #0
0877- A2 00 1790 LDX #0
0879- B1 00 1800 .7 LDA (MSG.PNTR),Y
087B- F0 24 1810 BEQ .9 ...END OF STRING
087D- 09 80 1820 ORA #80
087F- C9 A0 1830 CMP #A0 PRINTING CHARACTER
0881- B0 0A 1840 BCS .8 ...YES, GO PRINT IT
0883- 09 40 1850 ORA #40 ...NO, CONTROL, CHANGE TO
0885- 48 1860 PHA PRINTING FORM
0886- A9 DE 1870 LDA #" " PRINT #" " FOLLOWED BY CHAR
0888- E8 1880 INX
0889- 20 ED FD 1890 JSR COUT
088C- 68 1900 PLA
088D- 20 ED FD 1910 .8 JSR COUT
0890- E8 1920 INX
0891- 20 A7 08 1930 JSR INC ADVANCE MSG.PNTR
0894- E0 37 1940 CPX #55 IS THIS LINE FULL?
0896- 90 E1 1950 BCC .7 ...NO, KEEP GOING
0898- A2 18 1960 LDX #24 ...YES, START NEW LINE
089A- 20 CA 08 1970 JSR MARGIN INDENT
089D- A2 00 1980 LDX #0
089F- F0 D8 1990 BEQ .7 ...ALWAYS
2000 *-----
08A1- 20 8E FD 2010 .9 JSR CROUT
08A4- 4C 08 08 2020 JMP .1
2030 *-----
08A7- E6 00 2040 INC INC MSG.PNTR
08A9- D0 02 2050 BNE .1
08AB- E6 01 2060 INC MSG.PNTR+1
08AD- 60 2070 .1 RTS
2080 *-----
08AE- AD 00 C0 2090 PAUSE LDA KEYBOARD ANY KEY PRESSED?
08B1- 10 16 2100 BPL .3 ...NO, RETURN
08B3- 8D 10 C0 2110 STA STROBE ...YES, CLEAR STROBE
08B6- C9 8D 2120 CMP #8D WAS KEY <RETURN>?
08B8- D0 03 2130 BNE .2 ...NO, JUST A PAUSE
08BA- 4C D0 03 2140 .1 JMP $3D0 ...YES, ABORT
08BD- AD 00 C0 2150 LDA KEYBOARD ANY KEY PRESSED?
08C0- 10 FB 2160 BPL .2 ...NO, KEEP WAITING
08C2- 8D 10 C0 2170 STA STROBE ...YES, CLEAR STROBE
08C5- C9 8D 2180 CMP #8D WAS KEY <RETURN>?
08C7- F0 F1 2190 BEQ .1 ...YES, ABORT
08C9- 60 2200 .3 RTS ...NO, END OF PAUSE
2210 *-----
08CA- 20 8E FD 2220 MARGIN JSR CROUT START A NEW LINE
08CD- A9 A0 2230 LDA #A0 SKIP OVER (X) SPACES
08CF- 20 ED FD 2240 .10 JSR COUT
08D2- CA 2250 DEX
08D3- D0 FA 2260 BNE .10
08D5- 60 2270 RTS

```

Peeking at the CATALOG.....Bob Sander-Cederlof

Have you ever wanted just a quick peek at the catalog entry for a file? Maybe you want to know where the track/sector list is? Or maybe you want to see if there are any control characters in the name? Or if the number of sectors is more than 255? You need to peek, because CATALOG won't tell you these details.

After all these years, I found out a simple way to do it. That is, assuming you can OPEN, SAVE, LOCK, or otherwise somehow make DOS go looking for the file.

After DOS has found the file, it leaves the directory sector containing the filename in the buffer at \$B4BB-B5BA. DOS also leaves an index to the very byte at which the information on your file is found. The value in \$B39C, if added to the address \$B4C6, gives you the address of the start of the entry. \$22 bytes later it ends.

A minute ago I saved the contents of this and a few other short articles on a file named V4N5 SHORT SUBJECTS. Then I left my word processor, typed CALL -151 to get into the monitor, and... Well, here, look for yourself:

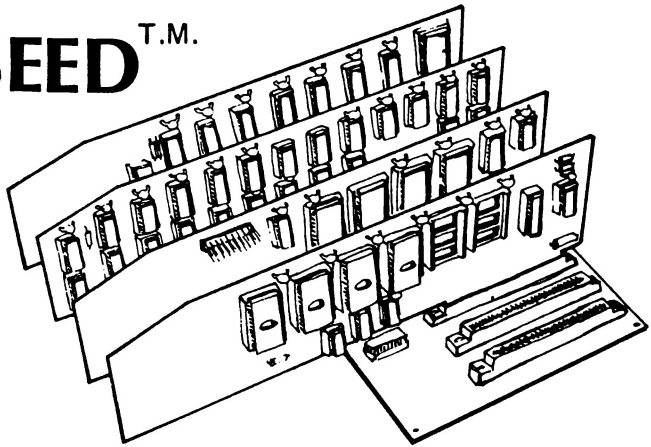
```
]CALL-151
*B39C
B39C- D2          (offset from B4C6)
*C6+D2
=98              (first byte of entry)
*98+22
=BA              (last byte of entry)
*B598.B5BA
B598- 0C 0E 00 D6 B4 CE B5 A0
B5A0- D3 C8 CF D2 D4 A0 D3 D5
B5A8- C2 CA C5 C3 D4 C3 A0 A0
B5B0- A0 A0 A0 A0 A0 A0 A0 A0
B5B8- A0 07 00
```

The first byte at B598 is the track, and the second is the sector, where the track/sector list for this file is stored. The third byte is the file type (00 means an unlocked text file). The last two bytes are the file size. All the bytes in between are the file name.

If you are interested in the entry for a file you cannot reach directly, perhaps because there are hidden characters in the name, just LOCK, UNLOCK, or whatever a file above or below it in the catalog. Then peek at B39C and B4BB...B5BA to find the entry you are really interested in.

We also took advantage of the fact that the track/sector list of a file read or written on can be found at the beginning of the file buffer. If there are three buffers (MAXFILES=3), and if the file in question was the only one being accessed at the time, the T/S list will be found at \$9600...\$96FF. You can get the data you need immediately, without even finding your favorite ZAP utility.

# APPLESEED<sup>T.M.</sup>



Appleseed is a complete computer system. It is designed using the bus conventions established by Apple Computer for the Apple II. Appleseed is designed as an alternative to using a full Apple II computer system. The Appleseed product line includes more than a dozen items including CPU, RAM, EPROM, UART, UNIVERSAL Boards as well as a number of other compatible items. This ad will highlight the Mother board.

## BX-DE-12 MOTHER BOARD

The BX-DE-12 Mother board is designed to be fully compatible with all of the Apple conventions. Ten card slots are provided. Seven of the slots are numbered in conformance with Apple standards. The additional three slots, lettered A, B and C, are used for boards which don't require a specific slot number. The CPU, RAM and EPROM boards are often placed in the slots A, B and C.

The main emphasis of the Appleseed system is illustrated by the Mother Board. The absolute minimum amount of circuitry is placed on the Mother Board; only the four ICs which are required for card slot selection are on the mother board. This approach helps in packaging (flexibility & smaller size), cost (buy only what you need) and repairability (isolate and fix problems through board substitution).

Appleseed products are made for O.E.M.s and serious industrial/scientific users. Send for literature on the full line of Appleseed products; and, watch here, each month, for additional items in the Appleseed line.

Appleseed products are not sold through computer stores.

Order direct from our plant in California.

Apple is a registered trademark of Apple Computer, Inc.

**DOUGLAS ELECTRONICS**

718 Marina Blvd., San Leandro, CA 94577 • (415) 483-8770

Clarification about our copyrights.....Bob Sander-Cederlof

We frequently are asked if it is all right to use ideas and even programs published in the Apple Assembly Line in articles or books our readers write for publication elsewhere, or even in software they plan to sell.

Sure! Just give us credit. Say where you got it, and hopefully tell your customers how they too can subscribe. The more you sell, the more we sell. The more we spread the good information around, the more we all benefit.

Fast Scroll for //e 80-column.....Bob Sander-Cederlof

The //e 80-column firmware scrolls in an annoying fashion. If you are trying to watch a listing go by, it looks like a bunch of kids on the playground, jumping up and down. And it is slower than almost any brand of 80-column card that plugs into slot 3.

The "slot 3" kind of 80-column card usually has a general purpose CRT controller chip on it. These chips use a wrap-around memory, and have one register that tells the chip where in memory to start the screen display. Scrolling is instantaneous, because it only involves writing a new address into two registers.

The //e 80-column card has no built-in features at all. All it is, is plain old RAM. A few extra circuits allow alternate columns to be taken first from the mother board and then from the 80-column card, back and forth. And the video rate is doubled, so 80 columns appear on each line. The scroll routine moves the whole screen up in two steps. First all the odd columns (in main memory) are moved up, and then all the even columns (in 80-column card memory). That is why you see the zig-zag effect.

The scroll is slower than a 40-column scroll by a factor of two. After all, it is essentially the same code, just called twice.

As I said in my article on fast scrolling in the September 1982 issue of AAL, you have to bear in mind that the authors of the programs in Apple ROM were not usually aiming for speed. They were trying to squeeze as much as possible into that tiny space, and make it as general as they could. The //e 80-column firmware supports windows smaller than a full screen, and that is seldom found in other types of 80-column cards.

On the other hand, since I am used to not having nice windows in the other cards, I can live with that in the //e. And I am having a hard time adjusting to that see-saw slow-motion scroller.

So, I re-wrote all the fast screen tricks from the September 1982 article to work in the //e with the Apple 80-column card. It scrolls as smooth as glass, but I still can't read it: now it's too fast!



```

1000 * S.SCREEN TRICKS //E 80-COLUMN
1010 *-----
1020 * FAST SCREEN CLEAR SUBROUTINE
1030 *-----
0800- A9 FF 1040 GCLEAR LDA #255
0802- 2C 1050 .HS 2C SKIP OVER NEXT TWO BYTES
0803- A9 A0 1060 CLEAR LDA #A0
0805- A0 77 1070 LDY #19
0807- A2 01 1080 SET LDX #1
0809- 9D 54 C0 1090 .2 STA $C054,X
080C- 99 00 04 1100 STA $400,Y LINES: 0 8 16
080F- 99 00 05 1110 STA $500,Y 2 10 18
0812- 99 00 06 1120 STA $600,Y 4 12 20
0815- 99 00 07 1130 STA $700,Y 6 14 22
0818- 99 80 04 1140 STA $480,Y 1 9 17
081B- 99 80 05 1150 STA $580,Y 3 11 19
081E- 99 80 06 1160 STA $680,Y 5 13 21
0821- 99 80 07 1170 STA $780,Y 7 15 23
0824- CA 1180 DEX
0825- 10 E2 1190 BPL .2
0827- 88 1200 DEY
0828- 10 DD 1210 BPL .1
082A- 60 1220 RTS
1230 *-----
1240 * SET SCREEN TO ALL VALUES
1250 *-----
082B- A9 00 1260 SETALL LDA #0
082D- 48 1270 .1 PHA
082E- 20 05 08 1280 JSR SET
0831- 68 1290 PLA
0832- 18 1300 CLC
0833- 69 01 1310 ADC #1
0835- D0 F6 1320 BNE .1
0837- 60 1330 RTS
1340 *-----
1350 * ALTERNATE SCREEN UNTIL KEY PRESSED
1360 *-----
0838- A9 20 1370 ALTER LDA #20 INVERSE BLANK
083A- 20 05 08 1380 JSR SET
083D- 20 03 08 1390 JSR CLEAR
0840- AD 00 C0 1400 LDA $C000
0843- 10 F3 1410 BPL ALTER
0845- 8D 10 C0 1420 STA $C010
0848- 60 1430 RTS
1440 *-----
1450 * FAST SCROLL UP SUBROUTINE
1460 *-----
0849- A0 00 1470 SCROLL LDY #0
084B- A2 01 1480 .1 LDX #1
084D- BD 54 C0 1490 .3 LDA $C054,X
0850- B9 00 04 1500 LDA $400,Y SAVE LINES: 0 8 16
0853- 48 1510 PHA
0854- B9 80 04 1520 LDA $480,Y MOVE 1>0, 9>8, 17>16
0857- 99 00 04 1530 STA $400,Y
085A- B9 00 05 1540 LDA $500,Y MOVE 2>1, 10>9, 18>17
085D- 99 80 04 1550 STA $480,Y
0860- B9 80 05 1560 LDA $580,Y MOVE 3>2, 11>10, 19>18
0863- 99 00 05 1570 STA $500,Y
0866- B9 00 06 1580 LDA $600,Y MOVE 4>3, 12>11, 20>19
0869- 99 80 05 1590 STA $580,Y
086C- B9 80 06 1600 LDA $680,Y ET CETERA
086F- 99 00 06 1610 STA $600,Y
0872- B9 00 07 1620 LDA $700,Y
0875- 99 80 06 1630 STA $680,Y
0878- B9 80 07 1640 LDA $780,Y
087B- 99 00 07 1650 STA $700,Y
087E- 68 1660 PLA MOVE 8>7, 16>15
087F- C0 28 1670 CPY #40
0881- 90 03 1680 BCC .2 DISCARD OLD LINE 0
0883- 99 58 07 1690 STA $780-40,Y
0886- CA 1700 .2 DEX
0887- 10 C4 1710 BPL .3
0889- C8 1720 INY
088A- C0 78 1730 CPY #120
088C- 90 BD 1740 BCC .1
088E- 60 1750 RTS

```

```

1760 *-----
1770 * SCROLL AROUND, MOVING TOP LINE TO BOTTOM
1780 *-----
088F- A0 27 1790 SCR LDY #39 SAVE TOP LINE ON STACK
0891- AD 54 C0 1800 .1 LDA $C054
0894- B9 00 04 1810 LDA $400,Y
0897- 48 1820 PHA
0898- AD 55 C0 1830 LDA $C055
089B- B9 00 04 1840 LDA $400,Y
089E- 48 1850 PHA
089F- 88 1860 DEY
08A0- 10 EF 1870 BPL .1
08A2- 20 49 08 1880 JSR SCROLL SCROLL SCREEN UP ONE LINE
08A5- A0 00 1890 LDY #0 STORE OLD TOP LINE
08A7- AD 55 C0 1900 .2 LDA $C055
08AA- 68 1910 PLA ON BOTTOM OF SCREEN
08AB- 99 D0 07 1920 STA $7D0,Y
08AE- AD 54 C0 1930 LDA $C054
08B1- 68 1940 PLA
08B2- 99 D0 07 1950 STA $7D0,Y
08B5- C8 1960 INY
08B6- C0 28 1970 CPY #40
08B8- 90 ED 1980 BCC .2
08BA- 60 1990 RTS
2000 *-----
2010 * ROTATE SCREEN UNTIL KEY PRESSED
2020 *-----
08BB- 20 8F 08 2030 S JSR SCR SCROLL AROUND ONCE
08BE- AD 00 C0 2040 LDA $C000 ANY KEY PRESSED?
08C1- 10 F8 2050 BPL $ NO, SCROLL AGAIN
08C3- 8D 10 C0 2060 STA $C010 YES, CLEAR STROBE
08C6- 60 2070 RTS ...AND RETURN

```

### DOS 3.3 Checksummer Debate Update.....Bob Sander-Cederlof

A letter from Bill Basham (Diversi-DOS author) defending the practice of omitting the automatic VERIFY after SAVE to gain speed, was published in the September 1983 Softalk (page 37, 38). At the top of page 38 Bill claimed that the checksumming method used by DOS was of no value at all, because the checksum only depended on the last two bytes. In other words, Bill claims that errors in the first 340 bytes of a sector will not be caught.

Diversi-DOS is a fine product, and many thousands are enjoying its advantages. Nevertheless, Bill is wrong about the checksum. It does indeed catch errors throughout a sector. For a complete explanation, see the February 1984 Softalk. David Wagner clearly explains how the checksummer works, and refutes Bill's claim. See his letter on page 40.

You can look at the code, too. We printed a full commented source listing of this code in the June 1981 issue of AAL.

So That's a Macintosh!.....Bill Morgan

Well, now we know. The rumors were basically correct: 68000 processor, 128K RAM, 3.5 inch disk drive (but only one), portable, Lisa descendant, about \$2500, and no expansion slots.

That last "feature" still has me a little shaken. I thought that if anybody knew better, it would be Apple, whose whole fortune is based on the expandability of the Apple ][. My first reaction was totally negative: who wants to bother with a dead-end machine? A total of 128K of RAM, and the screen memory occupies over 20K. Now that I've read a little more about the internals, and about the design objectives, things look a lot brighter. The on-board memory will be expandable to 512K when the 256K chips get more affordable.

System expansion will take place via the high-speed RS-422 serial ports. One of the designers pointed out that at 1 million bits per second (which can be reached with external clocking) you can transfer the entire memory image of the machine in one second. A couple of manufacturers (Davong and Tecmar) have already announced hard disks. Tecmar also announced an IEEE 488 interface. Macintosh designers also speak of "virtual slot" protocols for the serial ports, and "multi-drop (party line) capability".

There's another departure from usual Apple practice: no programming language is resident in the machine, or included in the purchase price! Several options will be available, including Pascal, Mac Basic, Microsoft Basic, Logo, and an Assembler/Debugger. The prices for the above packages will run in the \$100-\$150 range, not too bad. One article also mentioned C, about six months from now. It wasn't clear whether that was from Apple or an outside vendor. All of the above languages are scheduled for release in the next few months, except for Microsoft Basic. Russ Weaver, at Simtec/Quest, tells me he received that yesterday.

There is also 64K ROM (two 23256's) in the Mac, which holds the key to most programming. That ROM contains the code to support the "desk top" environment of mouse, icons, etc., the disk I/O, and the serial I/O. That is supposed to be 64K of the most tightly coded 68000 machine language around (as opposed to Lisa's compiled Pascal operating system code). I am told that there are over 400 entry points available to the programmer, with complete documentation coming soon from Apple for \$250.

Several information sources have already popped up. If you haven't seen the February issue of Byte, go get it. There is a large section on Mac, including the best technical data so far. There are already two magazines specializing in Macintosh: Macworld, from the publishers of PC World, and ST.Mac, from Softalk. (Saint Mac? Come on.) Macworld looks very good, especially for evaluation and demonstration of software. I haven't seen a copy of ST.Mac yet, but Softalk is about the best of the "general" Apple magazines so I expect good things from their entry. You can pay \$2495 for a Macintosh serial number and get a year's free subscription to ST.Mac.

Reminder about Wrap-Around Addressing.....Bill Parker

Buried on the right side of page 65 of the November, 1983 issue of Call APPLE is the examination by Martin Smith of another quirk of the 6502. I say "another quirk" because it is similar to the JMP indirect wrap-around bug. Remember it?

As reported in the October 1980 issue of Apple Assembly Line, "JMP (\$xxFF)" will not jump to the address pointed to by the two bytes beginning at \$xxFF; rather the two bytes at \$xxFF and \$xx00 will be used. (Where xx means any page of memory.)

A similar wrap-around situation can be found when indexing like this:

```
STACK .EQ $100
LDX #1
LDA STACK-1,X
```

Since STACK-1 is \$FF, a page zero address mode is assembled. Indexing from within page zero never leaves page zero, so the above example references location \$0000 rather than \$0100.

The above is important, because many programmers use it in a "WHEREAMI" section of code to find the program's current address:

```
STACK .EQ $100
WHEREAMI JSR $FF58 (known RTS instruction)
TSX
LDA STACK-1,X get PCL
LDY STACK,X get PCH
```

For the Merlin Assembler, the problem can be corrected by forcing the assembler to use an absolute addressing mode rather than a page zero addressing mode. This is done by suffixing a ":" to the opcode, like this:

```
LDA: STACK-1,X
```

The S-C Assemblers have no syntactical way to force absolute mode, but it can be done by defining the symbol STACK after its use. Here's an interesting example:

```
0800- BD FF 00 1000 LDA STACK-1,X
0100-          1010 STACK .EQ $100
0803- B5 FF 1020 LDA STACK-1,X
```

Since the assembler doesn't know the value of STACK in the first line, it has to assume it will be a two-byte address, and allocate that much space. By the time it gets to the last line it knows better.

The fact that indexing wraps around inside page zero is a plus sometimes. (I guess that explains why the chip works that way!) It has the effect of letting you use both positive and negative index offsets. Just beware of getting so used to negative offsets that you try to use them OUTSIDE page zero!

----- APPLE SOFTWARE -----

**NEW!!! FONT DOWNLOADER & EDITOR (\$39.00)**

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Can be used with every word processor capable of sending ESC and control codes to the printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed, emphasized, underlined, etc.) apply to custom fonts. Full HIRES screen editor lets you create your own custom characters and special graphics symbols. Compatible with many 'dumb' & 'smart' printer I/F cards. User driver option provided. Specify printer: Apple Dot Matrix Printer, C.Itoh 8510A (Prowriter), Epson FX-80/100 or OkiData 92/93.

**DISASM 2.2e - AN INTELLIGENT DISASSEMBLER (\$30.00)**

Investigate the inner workings of machine language programs. DISASM converts 6502 machine code into meaningful, symbolic source. Creates a standard DOS 3.3 text file which is directly compatible with DOS ToolKit, LISA and S-C (4.0 and MACROS) assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor & Pg Zero pg names included.) An address-based cross reference table provides further insight into the inner workings of machine language programs. DISASM is an invaluable machine language learning aid to both the novice & expert alike. **SOURCE code: \$60.00**

**S-C ASSEMBLER (Ver4.0 only) SUPPORT UTILITY PACKAGE (\$30.00)**

\* SC.XREF - Generates a GLOBAL LABEL Cross Reference Table for complete documentation of source listings. Formatting control accommodates all printer widths for best hardcopy outputs. \* SC.GSR - Global Search and Replace eliminates tedious manual renaming of labels. Search all or part of source. Optional prompting for user verification. \* SC.TAB - Tabulates source files into neat, readable form. **SOURCE code: \$40.00**

----- HARDWARE/FIRMWARE -----

**THE 'PERFORMER' CARD (\$39.00)**

Plugs into any Apple slot to convert your 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu provides easy access to printer fonts. Eliminates need to remember complicated ESC codes and key them in to setup printer. Added features include perforation skip, auto page numbering with date & title. Also includes large HIRES graphics screen dump in normal or inverse plus full page TEXT screen dump. Specify printer: Epson MX-80 with Graftrax-80, MX-100, MX-80/100 with GraftraxPlus, NEC 80923A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. Oki bonus: print EMPHASIZED & DOUBLE STRIKE fonts! **SOURCE code: \$30.00**

**FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM (\$25.00)**

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Three basic modes: Dumb Terminal, Remote Console & Programmable Modem. Added features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back option and built-in printer buffer. Supports most 80-column displays and the 1-wire shift key mod. Uses a superset of Apple's Comm card and Micromodem II commands. A-C hardware differences prevent 100% compatibility with Comm card. **SOURCE code: \$60.00**

**RAM/ROM DEVELOPMENT BOARD (\$30.00)**

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip. Use a 6116 type RAM chip for program development or just extra memory. Plug in a preprogrammed 2716 EPROM to keep your favorite routines 'on-line'. A versatile board with many uses! Maps into \$C000-CnFF & \$C800-CFFF memory space. Circuit diagram included.

**NEW!!! SINGLE BOARD COMPUTER KIT (\$20.00)**

Kit includes etched PC board (with solder mask and plated thru holes) and assembly instructions. User provides 6502 CPU, 6116 2K RAM, 6821 dual 8-bit I/O and 2732 4K EPROM plus misc common parts. Originally designed as intelligent printer interface - easily adapted to many applications needing dedicated controller. (Assembled and tested: \$119.00)

All assembly language SOURCE code is fully commented & provided in both S-C Assembler & standard TEXT formats on an Apple DOS 3.3 diskette. Specify your system configuration with order. Avoid a \$3.00 postage and handling charge by enclosing full payment with order (MasterCard & VISA excluded). Ask about our products for the VIC-20 and Commodore 64!

R A K - W A R E

41 Ralph Road West Orange NJ 07052 (201) 325-1885

Delays, delays, delays.....Bob Sander-Cederlof

We always want speed. Making computers compute faster keeps our industry humming. Yet nearly every major program has pieces of code called delays.

We use them to generate carefully controlled, timed events: for example, generating a musical tone. We use them to synchronize events, or to provide time for external events to occur. We even use them just to slow the computer down so we can watch it work.

Delays are used so often that Woz had the foresight to put a general purpose delay subroutine permanently inside the monitor ROM. It resides at \$FCA8. It is short (only 12 bytes), sweet (only uses one register, the same one which controls how long a delay you get), and slow (on purpose). Here is a listing:

WAIT	SEC	PREPARE TO SUBTRACT
.1	PHA	SAVE A COPY OF A-REG
.2	SBC #1	COUNT A-REG DOWN TO ZERO
	BNE .2	...UNTIL A=0
	PLA	GET SAVED COPY OF A-REG
	SBC #1	COUNT THIS COPY DOWN TOO
	BNE .1	...UNTIL A=0
	RTS	

To use this subroutine, you load the A-register with a value which will determine the length of the delay, and then JSR WAIT. When the subroutine returns, A=0 and somewhere between 29 and 167309 clock cycles have elapsed. The formula, somewhat confusingly printed on page 165 of the white Apple II Reference Manual (and elsewhere in other manuals), is:

$$\# \text{ cycles} = (5 * A * A + 27 * A + 26) / 2$$

For an example of its use, look in the monitor listing at \$FBDD (the bell routine). Examples of other timing loops are found in the tape cassette I/O routines (\$FCC9-\$FD0B) and the paddle reading subroutine (\$FB1E).

Bill and I spent the last two weeks working with software which surrounds the Novation Cat Modem. It is loaded with calls on the monitor WAIT subroutine. It is exceedingly tiresome to crank out a formula like the quadratic above by hand, or even with a calculator, over and over and over, when you have several Apples sitting in the same room!

After four or five trips to the manual and the calculator, I decided to work out the times for all possible values of the A-register. Once and for all.

Here is a little Applesoft program which does the job, and elsewhere in this AAL you will find a full page showing all the cycle counts.

```

1000 REM $FCA8 DELAY TIMES
1005 DIM P$(256)
1010 BL$ = " ":BR$ = " ":FOR A = 1 TO 256
1015 IF A = 65 THEN BL$ = " "
1016 IF A = 193 THEN BR$ = " "
1020 T = (A * A * 5 + A * 27 + 26) / 2
1030 X = A: IF A = 256 THEN X = 0
1040 X$ = RIGHT$( " " + STR$(X),3)
1050 TR$ = RIGHT$( ( STR$(T + 1000000),3):T = INT (T / 1000):TL$ = RIGHT$(
  (" " + STR$(T),3):T$ = TL$ + "." + TR$
1060 H = INT (X / 16):L = X - H * 16
1070 H = H + 7 * (H > 9):L = L + 7 * (L > 9)
1080 H$ = "$" + CHR$(H + 48) + CHR$(L + 48)
1090 P$(A) = H$ + BL$ + X$ + BR$ + T$
1100 NEXT
2000 FOR I = 1 TO 64: PRINT P$(I) " " P$(I + 64) " " P$(I + 128) " "
  P$(I + 192): NEXT

```

The A-register values are given in both hex and decimal. The delay count is given in thousands of cycles. Each cycle is close to one microsecond, so you could think of the counts as being in milliseconds.

The purists among you will want to multiply these cycle counts by the ACTUAL clock period (.9799268644 microseconds average, according to Sather) to get ACTUAL time.

RWTS in DOS or ProDOS also give lessons in the use of precise delays. You will find weird little pieces of code which make no sense whatever inside RWTS. Things like PHA followed immediately by PLA, followed by a NOP. These are usually just delaying tricks. A PHA-PLA pair takes exactly seven cycles, a NOP 2 more. There is a delay while waiting for the motor to come up to speed. Another while stepping the head from track to track.

These last two are intertwined, so that delays used while stepping across tracks count towards the total delay required to get the disk rotating at 300 rpm.

Don Lancaster in his Enhancing the Apple books makes good use of delays in synchronizing graphics generation with the CRT. By updating a picture in one graphics page while displaying another, and then switching pages, you can get pretty impressive animation. However, the page flipping operations sometimes splatter the display. Using delays just right, you can make the switching occur when it won't be noticed. You can even mix graphics into the middle of a text screen or vice versa, or mix hi-res and lo-res on the same screen.

Jim Sather in "Understanding the Apple II" also uses delays to control the screen switches in interesting ways. Jim figured out exactly how many cycles everything in the video generation circuitry takes. Using his programs you can even use hi-res to draw underlines on text screens! A horizontal scan takes exactly 65 clock cycles. A vertical scan takes exactly 17030 cycles. The following program, adapted from one given on page 3-16 of Sather's book, splits the screen between hi-res and lo-res. Tapping the space bar moves the boundaries of the split. Play with it!

```

1000 *SAVE SATHER 3-16
1010 *-----
1020 *      HIRES-LORES SPLIT
1030 *      SATHER 3-16
1040 *-----
C000- 1050 KYBD      .EQ $C000
C010- 1060 STRB     .EQ $C010
C050- 1070 GRAPHICS .EQ $C050
C051- 1080 TEXT     .EQ $C051
C052- 1090 NOTMIXED .EQ $C052
C054- 1100 PAGE1    .EQ $C054
C056- 1110 LORES    .EQ $C056
1120 *-----
1130 *      TOGGLE HI/LO-RES EVERY 8515 CYCLES
1140 *-----
1150      .OR $300
0300- AC 54 C0 1160 SPLIT LDY PAGE1 HI/LO PAGE 1
0303- AC 52 C0 1170 LDY NOTMIXED
0306- AC 50 C0 1180 LDY GRAPHICS
1190 *-----
0309- A0 27 1200 SLEW LDY #39 (2) SLEW SCREEN IF KEY PRESSED
030B- 20 2B 03 1210 JSR WAITX10 (390) 6*65+7 CYCLES
030E- AC 10 C0 1220 LDY STRB (4)
1230 *-----
0311- AC 00 C0 1240 KEYCHK LDY KYBD (4) ANY KEY PRESSED?
0314- 30 F3 1250 BMI SLEW (2 OR 3) YES, SLEW ONE LINE
0316- 69 01 1260 ADC #1 (2) MAKE ALTERNATING 0 AND 1
0318- 29 01 1270 AND #1 (2)
031A- AA 1280 TAX (2) REMEMBER, 0 OR 1
031B- BC 56 C0 1290 LDY LORES,X (4) LORES IF X=0, HIRES IF X=1
031E- A2 08 1300 LDX #8 (2)
0320- 20 37 03 1310 JSR WAITX1K (8000)
0323- A0 31 1320 LDY #49 (2)
0325- 20 2B 03 1330 JSR WAITX10 (490)
0328- 18 1340 CLC (2)
0329- 90 E6 1350 BCC KEYCHK (3) ...ALWAYS
1360 *      =====
1370 *      (8515)
1380 *-----
1390 *-----
1400 *      TIMING ROUTINES
1410 *-----
1420 *-----
1430 *---WAIT 10Y CYCLES-----
1440 *---(INCLUDING JSR)-----
032B- 88 1450 WAITX10 DEY (2) WAIT Y-REG TIMES 10
032C- 88 1460 .1 DEY (2)
032D- EA 1470 NOP (2)
032E- D0 01 1480 BNE .2 (3 OR 2)
0330- 60 1490 RTS (6)
0331- D0 F9 1500 .2 BNE .1 (3) ...ALWAYS
1510 *-----
1520 *-----
1530 *---WAIT 1000X CYCLES-----
1540 *---(INCLUDING JSR)-----
0333- 48 1550 LOOP1K PHA (3)
0334- 68 1560 PLA (4)
0335- EA 1570 NOP (2)
0336- EA 1580 NOP (2)
0337- A0 62 1590 WAITX1K LDY #98 (2) WAIT X-REG TIMES 1000
0339- 20 2B 03 1600 JSR WAITX10 (980)
033C- EA 1610 NOP (2)
033D- CA 1620 DEX (2)
033E- D0 F3 1630 BNE LOOP1K (3 OR 2)
0340- 60 1640 RTS (6)
1650 *-----

```



```

1660 *-----
1670 * HORIZONTAL SPLIT
1680 * BY BOB SANDER-CEDERLOF
1690 *-----
1700 HSPLIT
0341- AD 50 C0 1710 LDA GRAPHICS (4 4)
0344- AD 00 C0 1720 LDA KYBD (4 4)
0347- 10 05 1730 BPL .1 (3 2)
0349- 8D 10 C0 1740 STA STRB (4 3)
034C- 30 03 1750 BMI .3 (3)
034E- EA 1760 .1 NOP (2)
034F- 10 00 1770 BPL .3 (3)
0351- 20 64 03 1780 .3 JSR DLY12 (12 12)
0354- EA 1790 NOP (2 2)
0355- EA 1800 NOP (2 2)
1810 *
1820 *
1830 *
0356- AD 51 C0 1840 LDA TEXT (4)
0359- 20 61 03 1850 JSR DLY21 (21)
035C- 18 1860 CLC (2)
035D- 90 00 1870 BCC .2 (3)
035F- 90 E0 1880 BCC HSPLIT (3)
1890 *
1900 *
1910 *
1920 *
1930 *-----
0361- 48 1930 JSR DLY.. (6) (6)
0362- 68 1940 DLY21 PHA (3)
0363- EA 1950 PLA (4)
0364- 60 1960 NOP (2)
1970 DLY12 RTS (6) (6)
1980 *
1990 *
(21) (12)

```

SEE IF SHOULD SLEW

## Complete, Commented Source Code!

Our software is not only unlocked and fully copyable

...we often provide the complete source code on disk, at unbelievable prices!

**S-C Macro Assembler.** The key to unlocking all the mysteries of machine language. Combined editor/assembler with 29 commands, 20 directives. Macros, conditional assembly, global replace, edit, and more. Highest rating "The Book of Apple Software" in 1983 and 1984. \$80.

**Powerful cross-assembler modules** also available to owners of S-C Macro Assembler. You can develop software on your Apple for 6800, 6805, 6809, 68000, 8085, 8048, 8051, 1802, LSI-11, and Z-80 microprocessors. \$50 each.

**S-C Xref.** A support program which works with the S-C Macro Assembler to generate an alphabetized listing of all labels in a source file, showing with each label the line number where it is defined along with all line numbers containing references to the label. You get the complete source code for this amazingly fast program, on disk in format for S-C Macro Assembler. \$50.

**Full Screen Editor.** Integrates with the built-in line-oriented editor in the S-C Macro Assembler to provide a powerful full-screen editor for your assembly language source files. Drivers for Videx, STB80, and Apple //e 80-column boards are included, as well as standard 40-column version. Requires 64K RAM in your Apple. Complete source code on disk included. \$50.

**S-C Docu-Mentor for Applesoft.** Complete documentation of Applesoft internals. Using your ROM Applesoft, produces ready-to-assemble source code with full labels and comments. Educational, entertaining, and extremely helpful. Requires S-C Macro Assembler and two disk drives. \$50.

**S-C Word Processor.** The one we use for manuals, letters, our monthly newsletter, and whatever. 40-columns only, requires lower-case display and shiftkey mod. Works with standard DOS text files, but at super fast (100 sectors in 7 seconds). No competition to WordStar, but you get complete source code! \$50.

**Apple Assembly Line.** Monthly newsletter published since October, 1980, for assembly language programmers or those who would like to be. Tutorial articles, advanced techniques, handy utility programs, and commented listings of code in DOS, ProDOS, and the Apple ROMs. Helps you get the most out of your Apple! \$18/year.

**S-C SOFTWARE CORPORATION**  
2331 Gus Thomasson, Suite 125  
Dallas, TX 75228 (214) 324-2050

Professional Apple Software Since 1978  
Visa, MasterCard, American Express, COD accepted.

Apple is a trademark of Apple Computer Inc.



(A) Hex	(A) Decimal	Kilocycles	(A) Hex	(A) Decimal	Kilocycles	(A) Hex	(A) Decimal	Kilocycles	(A) Hex	(A) Decimal	Kilocycles
\$01	1	0.029	\$41	65	11.453	\$81	129	43.357	\$C1	193	95.741
\$02	2	0.050	\$42	66	11.794	\$82	130	44.018	\$C2	194	96.722
\$03	3	0.076	\$43	67	12.140	\$83	131	44.684	\$C3	195	97.708
\$04	4	0.107	\$44	68	12.491	\$84	132	45.355	\$C4	196	98.699
\$05	5	0.143	\$45	69	12.847	\$85	133	46.031	\$C5	197	99.695
\$06	6	0.184	\$46	70	13.208	\$86	134	46.711	\$C6	198	100.696
\$07	7	0.230	\$47	71	13.574	\$87	135	47.396	\$C7	199	101.702
\$08	8	0.281	\$48	72	13.945	\$88	136	48.089	\$C8	200	102.713
\$09	9	0.337	\$49	73	14.321	\$89	137	48.785	\$C9	201	103.729
\$0A	10	0.398	\$4A	74	14.702	\$8A	138	49.486	\$CA	202	104.750
\$0B	11	0.464	\$4B	75	15.088	\$8B	139	50.192	\$CB	203	105.776
\$0C	12	0.535	\$4C	76	15.479	\$8C	140	50.903	\$CC	204	106.807
\$0D	13	0.611	\$4D	77	15.875	\$8D	141	51.619	\$CD	205	107.843
\$0E	14	0.692	\$4E	78	16.276	\$8E	142	52.340	\$CE	206	108.884
\$0F	15	0.778	\$4F	79	16.682	\$8F	143	53.066	\$CF	207	109.930
\$10	16	0.869	\$50	80	17.093	\$90	144	53.797	\$D0	208	110.981
\$11	17	0.965	\$51	81	17.509	\$91	145	54.533	\$D1	209	112.037
\$12	18	1.066	\$52	82	17.930	\$92	146	55.274	\$D2	210	113.098
\$13	19	1.172	\$53	83	18.356	\$93	147	56.020	\$D3	211	114.164
\$14	20	1.283	\$54	84	18.787	\$94	148	56.771	\$D4	212	115.235
\$15	21	1.399	\$55	85	19.223	\$95	149	57.527	\$D5	213	116.311
\$16	22	1.520	\$56	86	19.664	\$96	150	58.288	\$D6	214	117.392
\$17	23	1.646	\$57	87	20.110	\$97	151	59.054	\$D7	215	118.478
\$18	24	1.777	\$58	88	20.561	\$98	152	59.825	\$D8	216	119.569
\$19	25	1.913	\$59	89	21.017	\$99	153	60.601	\$D9	217	120.665
\$1A	26	2.054	\$5A	90	21.478	\$9A	154	61.382	\$DA	218	121.766
\$1B	27	2.200	\$5B	91	21.944	\$9B	155	62.168	\$DB	219	122.872
\$1C	28	2.351	\$5C	92	22.415	\$9C	156	62.959	\$DC	220	123.983
\$1D	29	2.507	\$5D	93	22.891	\$9D	157	63.755	\$DD	221	125.099
\$1E	30	2.668	\$5E	94	23.372	\$9E	158	64.556	\$DE	222	126.220
\$1F	31	2.834	\$5F	95	23.858	\$9F	159	65.362	\$DF	223	127.346
\$20	32	3.005	\$60	96	24.349	\$A0	160	66.173	\$E0	224	128.477
\$21	33	3.181	\$61	97	24.845	\$A1	161	66.989	\$E1	225	129.613
\$22	34	3.362	\$62	98	25.346	\$A2	162	67.810	\$E2	226	130.754
\$23	35	3.548	\$63	99	25.852	\$A3	163	68.636	\$E3	227	131.900
\$24	36	3.739	\$64	100	26.363	\$A4	164	69.467	\$E4	228	133.051
\$25	37	3.935	\$65	101	26.879	\$A5	165	70.303	\$E5	229	134.207
\$26	38	4.136	\$66	102	27.400	\$A6	166	71.144	\$E6	230	135.368
\$27	39	4.342	\$67	103	27.926	\$A7	167	71.990	\$E7	231	136.534
\$28	40	4.553	\$68	104	28.457	\$A8	168	72.841	\$E8	232	137.705
\$29	41	4.769	\$69	105	28.993	\$A9	169	73.697	\$E9	233	138.881
\$2A	42	4.990	\$6A	106	29.534	\$AA	170	74.558	\$EA	234	140.062
\$2B	43	5.216	\$6B	107	30.080	\$AB	171	75.424	\$EB	235	141.248
\$2C	44	5.447	\$6C	108	30.631	\$AC	172	76.295	\$EC	236	142.439
\$2D	45	5.683	\$6D	109	31.187	\$AD	173	77.171	\$ED	237	143.635
\$2E	46	5.924	\$6E	110	31.748	\$AE	174	78.052	\$EE	238	144.836
\$2F	47	6.170	\$6F	111	32.314	\$AF	175	78.938	\$EF	239	146.042
\$30	48	6.421	\$70	112	32.885	\$B0	176	79.829	\$F0	240	147.253
\$31	49	6.677	\$71	113	33.461	\$B1	177	80.725	\$F1	241	148.469
\$32	50	6.938	\$72	114	34.042	\$B2	178	81.626	\$F2	242	149.690
\$33	51	7.204	\$73	115	34.628	\$B3	179	82.532	\$F3	243	150.916
\$34	52	7.475	\$74	116	35.219	\$B4	180	83.443	\$F4	244	152.147
\$35	53	7.751	\$75	117	35.815	\$B5	181	84.359	\$F5	245	153.383
\$36	54	8.032	\$76	118	36.416	\$B6	182	85.280	\$F6	246	154.624
\$37	55	8.318	\$77	119	37.022	\$B7	183	86.206	\$F7	247	155.870
\$38	56	8.609	\$78	120	37.633	\$B8	184	87.137	\$F8	248	157.121
\$39	57	8.905	\$79	121	38.249	\$B9	185	88.073	\$F9	249	158.377
\$3A	58	9.206	\$7A	122	38.870	\$BA	186	89.014	\$FA	250	159.638
\$3B	59	9.512	\$7B	123	39.496	\$BB	187	89.960	\$FB	251	160.904
\$3C	60	9.823	\$7C	124	40.127	\$BC	188	90.911	\$FC	252	162.175
\$3D	61	10.139	\$7D	125	40.763	\$BD	189	91.867	\$FD	253	163.451
\$3E	62	10.460	\$7E	126	41.404	\$BE	190	92.828	\$FE	254	164.732
\$3F	63	10.786	\$7F	127	42.050	\$BF	191	93.794	\$FF	255	166.018
\$40	64	11.117	\$80	128	42.701	\$C0	192	94.765	\$00	0	167.309

MON.WAIT (\$FCA8) Delay Times

Here is a quick look at some of the books and articles about the 68000 that I have found to be helpful.

Another possible source of 68000 information is the newsletter "DTACK Grounded", published by Digital Acoustics, 1415 E. McFadden, Suite F, Santa Ana, CA 92705. I've only seen one or two issues, back before I got interested in 68000, so I don't know exactly what they've been up to lately. I'll be finding out soon and pass it on. I might note that the issue I have (#7, Feb-Mar 1982) contains about 12 pages of more-or-less interesting gossip, and no code. I don't know if that is typical.

#### Books:

68000 Assembly Language Programming. Gerry Kane, Doug Hawkins & Lance Leventhal. OSBORNE/McGraw-Hill, 1981.

The Leventhal book. Need I say more? Recommended.

The 68000: Principles and Programming. Leo. J. Scanlon. Blacksburg/Sams, 1981.

Tutorial. Looks pretty good. Recommended.

MC68000 16-bit Microprocessor User's Manual, third edition. Motorola/Prentice-Hall, 1982.

Motorola's manual. THE basic reference. There is a fourth edition coming this year (1984). There is also a Mostek version of this book, but the Motorola edition is better.

MK68000 Microcomputer Programming Reference Guide. Mostek Corp, 1981.

A 42-page Quick Reference Card. Isn't that a bit much?

Programming the M68000. Tim King and Brian Knight. Addison-Wesley, 1983.

Tutorial. Looks very good. Lots of examples, building up to a simple monitor/debugger. Recommended.

#### Articles:

Design Philosophy Behind Motorola's MC68000. Thomas W. Starnes (of Motorola, Inc.) Byte. April-June, 1983 (3 parts).

Very good. Lives up to the title. Recommended.

68000 Instructions and Addressing Modes. Joe Hootman. Micro. #'s 52,54-57,60-62 (8 parts).

Summaries of the instruction set. OK if you already have the stack of Micro back issues.

An MC68000 Overview. Joe Jelemensky & Tom Whiteside. Micro. #'s 52,54.

Some good examples of the instructions at work.

Table of //e Soft Switches.....Bob Sander-Cederlof

For some reason none of the //e manuals I own give a complete chart in one place of all the new soft switches. If I print one here, I'll have one when I need it, so that's what the first chart on the following page is.

I have ordered them according to the location you peek at to find which position the soft switch is in. The first column is the location you read. The sense of the switch is given by bit 7 of the byte you read, and that bit's value is given at the top of the next two columns.

Note that there is an error in the Apple //e Reference Manual, on both pages 133 and 214, where the SLOTCXROM soft switch is described. In both places, the slot/internal designations are backwards. It looks like the book was written rationally, and the circuit behaves irrationally, because the SLOTC3ROM switch operates the opposite manner from the SLOTCXROM switch. Oh well...

The maze of information regarding the bank switching switches has me baffled. The second chart should help demystify things. I show which switches to throw which way to make any particular range of memory come from the main 64K or the auxiliary bank. To keep the chart from growing beyond the page, I did not include the LCBANK, SLOTCX, or SLOTC3 switches.

OBJ.APWRT][F

Detailed, complete, and thorough disassembly script includes full details on customizing Applewriter IIe and capturing your own source code for modification.

Two unlocked, jammed-full and double-sided diskettes plus a free "must have" bonus book for only \$29.95.

SOURCECODE  
% Synergetics  
Box 1300-AAL  
Thatcher, AZ 85552  
(602) 428-4073

VISA and MASTERCARGE accepted.

# Apple //e Soft Switches

Status	0	1
C011	C08(8-B)	C08(0-3)
LCBANK	Bank 1	Bank 2
D000-DFFF		
C012	C081,2,9,A	C080,3,8,B
LCRAM	Select ROM	Select RAM
D000-FFFF		
C013	C002	C003
RAMRD	Read Main	Read Aux
200-BFFF		
C014	C004	C005
RAMWRT	Write Main	Write Aux
200-BFFF		
C015	C006	C007
SLOT CX	Slot	Internal
C100-C7FF		
C016	C008	C009
ALTZP	Main	Aux
0-1FF, D000-FFFF		
C017	C00A	C00B
SLOT C3	Internal	Slot
C300-C3FF, C800-CFFF		
C018	C000	C001
80STORE	RAMRD/RAMWRT	PAGE2
400-7FF, 2000-3FFF		
C019		
VBL	in display	in blanking
C01A	C050	C051
TEXT	Graphics	Text
C01B	C052	C053
MIXED	All Text or all graphics	Mixed text & graphics
C01C	C054	C055
PAGE2	Page 1/Main	Page 2/Aux
400-7FF, 2000-3FFF		
C01D	C056	C057
HIRES	Lo-Res	Hi-Res

Status	0	1
C01E	C00E	C00F
CHARSET	Normal	Alternate
C01F	C00C	C00D
80COL	40 Columns	80 Columns
Address	Main Memory	Aux Memory
E000-FFFF	C008 ALTZP=0 C080,3,8,B LCRAM=1	C009 ALTZP=1 C080,3,8,B LCRAM=1
C000-CFFF	I/O Space	
4000-BFFF	Read: C002 Write: C004	Read: C003 Write: C005
2000-3FFF	C001 80STORE=1 C057 HIRES=1 C054 PAGE2=0	C001 80STORE=1 C057 HIRES=1 C055 PAGE2=1
or	C000 80STORE=0 Read: C002 Write: C004	C000 80STORE=0 Read: C003 Write: C005
or	C056 HIRES=0 Read: C002 Write: C004	C056 HIRES=0 Read: C003 Write: C005
800-1FFF	Read: C002 Write: C004	Read: C003 Write: C005
400-7FF	C001 80STORE=1 C054 PAGE2=0	C001 80STORE=1 C055 PAGE2=1
or	C000 80STORE=0 Read: C002 Write: C004	C000 80STORE=0 Read: C003 Write: C005
200-3FF	Read: C002 Write: C004	Read: C003 Write: C005
0-1FF	C008 ALTZP=0	C009 ALTZP=1

Good programs interact frequently with their users, providing error messages, helpful prompts, and information about what the program is doing. For programmers, this raises the question of what to do with the messages once they have been printed, especially if you want to get rid of them while leaving the rest of the screen intact.

I have used several strategies to clear specific areas of the text screen. The simplest solution, and probably the most commonly used, is to place all messages at the end of the page. Then you can HTAB and VTAB to the first character of the message and CALL the Monitor routine at \$FC42 (CLREOP). From Applesoft, CALL -958. Such messages must be kept to the lower part of the screen, however, and the method can interfere with decorative borders, etc., placed around your screens.

Another thing I have done is to print strings of blanks over the offending message. I use a loop to HTAB and VTAB to the left margin of the message area, incrementing the vertical coordinate each time, then printing a string variable set to a predetermined number of blanks. This method is slow, but not unbearable. Still, it is clumsy and wastes memory storing the blanks.

Of course, instantaneous clears of a given area are easily done by resetting the text window through POKEing values to locations \$25-\$28, then executing a HOME. This requires POKEing 4 values before the clear, however, and POKEing 4 coordinates to reset the current window when you are done (or "TEXT" to reset the default window). Downright unpleasant. For a time I resorted to this method to protect my decorative borders, however.

Now I have come up with a routine that I think is an improvement over the above. It clears rectangular areas of the text screen given the width and depth (number of lines) needed. Because it uses the Monitor COUT routine, it should also work with those hi-res character generator utilities that interface to the normal output hooks, giving a controlled hi-res screen clear. While it requires Applesoft in ROM, it is fully relocatable, making it ideal for people who use Ampersand utilities like AmperMagic or The Routine Machine.

The routine, which I call "ERASE", is used by first HTABing and VTABing to the upper left corner of the area to be cleared. Then CALL the routine giving the width and depth of the area to be cleared, using commas, like so:

```
CALL ADDRESS,WIDTH,DEPTH
```

For example, assume you BLOAD the routine at \$300, the most common place to do such things. (At least while we are testing the program.) Then, to clear an area 15 characters wide by 4 lines deep, write:

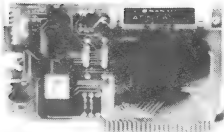
```
CALL 768,15,4
```

# Apple Peripherals Are All We Make

## That's Why We're So Good At It!

### THE NEW TIMEMASTER II

Automatically date stamps files with PRO-DOS



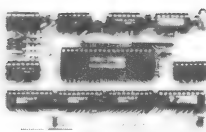
NEW 1984 DESIGN  
An official PRO-DOS Clock

- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- NiCad battery keeps the TIMEMASTER II running for over ten years.
- Full emulation of ALL other clocks. Yes, we emulate Brand A, Brand T, Brand P, Brand C, Brand S and Brand M too. It's easy for the TIMEMASTER II to emulate other clocks, we just drop off features. That's why we can emulate others, but others CANT emulate us.
- The TIMEMASTER II will automatically emulate the correct clock card for the software you're using. You can also give the TIMEMASTER II a simple command to tell it which clock to emulate (but you'll like the Timemaster mode better). This is great for writing programs for those poor unfortunates that bought some other clock card.
- Basic, Machine Code, CP/M and Pascal software on 2 disks!
- Eight software controlled interrupts so you can execute two programs at the same time (many examples are included).
- On-board timer lets you time any interval up to 48 days long down to the nearest millisecond.

The TIMEMASTER II includes 2 disks with some really fantastic time oriented programs (over 40) including appointment book so you'll never forget to do anything again. Enter your appointments up to a year in advance then forget them. Plus DOS date so it will automatically add the date when disk files are created or modified. The disk is over a \$200.00 value alone—we give the software others sell. All software packages for business, data base management and communications are made to read the TIMEMASTER II. If you want the most powerful and the easiest to use clock for your Apple, you want a TIMEMASTER II.

PRICE \$129.00

### Z-80 PLUS!

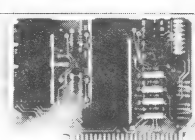
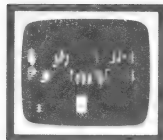


- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microsoft disks (no pre-boot required).
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the II+ and Franklin).
- Runs WORD STAR, dBASE II, COBOL-80, FORTRAN-80, PEACHTREE and ALL other CP/M software with no pre-boot.
- A semi-custom I.C. and a low parts count allows the Z-80 Plus to fly thru CP/M programs at a very low power level. (We use the Z-80A at fast 4MHz.)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

Don't confuse the Z-80 Plus with crude copies of the microsoft card. The Z-80 Plus employs a much more sophisticated and reliable design. With the Z-80 Plus you can access the largest body of software in existence. Two computers in one and the advantages of both, all at an unbelievably low price.

PRICE \$139.00

### Super Music Synthesizer Improved Hardware and Software



- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and the fastest music input system available anywhere.
- We give you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Full control of attack, volume, decay, sustain and release.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all our card's features. Their software sounds the same in our synthesizer.)
- Our card will play notes from 30HZ to beyond human hearing.
- Automatic shutoff on power-up or if reset is pushed.
- Many many more features.

PRICE \$159.00

### Viewmaster 80

There used to be about a dozen 80 column cards for the Apple, now there's only ONE.

- TOTALLY Vdex Compatible.
- 80 characters by 24 lines, with a sharp 7x9 dot matrix.
- On-board 40/80 soft video switch with manual 40 column override
- Fully compatible with ALL Apple languages and software—there are NO exceptions.
- Low power consumption through the use of CMOS devices.
- All connections are made with standard video connectors.
- Both upper and lower case characters are standard.
- All new design (using a new Microprocessor based C.R.T. controller) for a beautiful razor sharp display.
- The VIEWMASTER incorporates all the features of all other 80 column cards, plus many new improvements.

	PRICE	ONLY 80 COLUMNS	80/80 SWITCH	LOW POWER DESIGN	40 COLUMN SWITCH	7x9 DOT MATRIX	UPPER/LOWER CASE	80 COLUMN OVERRIDE	POWERFUL CONTROLLER
VIEWMASTER 179	YES	YES	YES	YES	YES	YES	YES	YES	YES
SUPRTERM	MORE	NO	YES	NO	NO	NO	NO	YES	YES
WIZARD80	MORE	NO	NO	NO	NO	YES	NO	YES	YES
VISIC80	MORE	YES	YES	NO	NO	NO	NO	NO	NO
OMNIVISION	MORE	NO	YES	NO	NO	NO	NO	YES	YES
VIEWMASTER	MORE	YES	YES	NO	NO	YES	NO	YES	YES
SMARTERM	MORE	YES	YES	NO	NO	NO	YES	YES	NO
VIDEOTERM	MORE	NO	NO	YES	NO	YES	YES	NO	YES

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, Easywriter, Apple Writer II, VisiCalc, and all others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE \$179.00

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- Compatible with all Apple IIe 80 column and extended 80 column card software (same physical size as Apple's 64K card).
- Can be used as a solid state disk drive to make your programs run up to 20 times FASTER (the 64K configuration will act as half a drive).
- Permits your IIe to use the new double high resolution graphics.
- Automatically expands VisiCalc to 95 K storage in 80 columns! The 64K config. is all that's needed, 128K can take you even higher.
- PRO-DOS will use the MemoryMaster IIe as a high speed disk drive.

### MemoryMaster IIe 128K RAM Card

- Precision software disk emulation for Basic, Pascal and CP/M is available at a very low cost. NOT copy protected.
- Documentation included, we show you how to use all 192K.

If you already have Apple's 64K card, just order the MEMORYMASTER IIe with 64K and use the 64K from your old board to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

MemoryMaster IIe with 128K	\$249
Upgradeable MemoryMaster IIe with 64K	\$169
Non-Upgradeable MemoryMaster IIe with 64K	\$149

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in the APPLE IIe, II+, II+ and Franklin. The MemoryMaster IIe is the only Applied Engineering also manufactures a full line of data acquisition and control products for the Apple, A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle THREE YEAR WARRANTY.

Texas Residents Add 5% Sales Tax  
Add \$10.00 if Outside U.S.A.  
Dealer Inquiries Welcome

Send Check or Money Order to:  
APPLIED ENGINEERING  
P.O. Box 798  
Carrollton, TX 75006

Call (214) 492-2027  
8 a.m. to 11 p.m. 7 days a week  
MasterCard, Visa & C.O.D. Welcome  
No extra charge for credit cards

The command shown above uses simple constants, but ERASE can handle any quantities "width" and "depth" up to formulas as complex as those Applesoft can normally handle. (I can't brag about that part, since all the work is done by Applesoft's formula evaluation routine "FRMEVL", called indirectly in my program by the "JSR GETBYT".

In case you don't have John Crossley's article on Applesoft Internal Entry Points, GETBYT is a subsidiary routine that evaluates formulas, bringing back a single-byte integer in the X-register and in location \$A1--"FACLO". I don't use "FACLO" in this routine. GETBYT gives an illegal quantity error if the formula evaluates to more than 255 or less than 0.)

If you specify a width or depth of zero, ERASE will give an illegal quantity error. If the width of the line goes past the right edge of the screen, the blanks will wrap around the screen on the next line down. ERASE will pick up at the correct horizontal/vertical location when clearing subsequent lines, however. If the area to be erased goes past the bottom of the screen, do not fear: ERASE wraps around to the top of the screen. Your program and variables will not be hurt.

Here is a short Applesoft program that demonstrates ERASE in action. The program first fills the entire screen with asterisks, and then clears three windows. The first window wraps around from the right edge of the screen to the left. The second wraps around from the bottom to the top. The third is in the middle of the screen. (I am assuming a 40-column screen here.)

```

100 FOR I = 1 TO 24: PRINT
    "*****"; : NEXT
110 HTAB 30: VTAB 10: CALL 768,20,5
120 HTAB 10: VTAB 20: CALL 768,20,8
130 HTAB 15: VTAB 10: CALL 768,10,5

```

And here is another demo, one which is closer to the way you will find yourself using ERASE. This one prints an array of six messages in six windows on the screen, and lets you selectively erase them in any order one-by-one. As it turned out, the way I located the upper corners of the messages involved some lengthy formulas, but these ended up in the HTAB and VTAB statements. Note that I could have used data statements for similar results.

```

90 PRINT CHR$(4);"BLOAD ERASE"
100 HOME : VTAB 5
110 PRINT "    AREA #1    AREA #2    AREA #3"
120 PRINT "    IN THIS    AROUND    LIVES IN"
130 PRINT "    VICINITY    HERE    THIS CORNER"
140 VTAB 12
150 PRINT "    AREA #4    AREA #5    AREA #6"
160 PRINT "    THAT'S    AT YOUR    IS ALSO"
170 PRINT "    ME!    SERVICE!    GREAT."
180 HTAB 15: VTAB 20
190 PRINT "ERASE WHICH?";: GET A$
200 ON ASC (A$) = 13 GOTO 260
210 A = VAL (A$): ON A < 1 OR A > 6 GOTO 180
220 HTAB 12 * ( INT (((A - 1) / 3) - INT ((A - 1) / 3)) * 3 + .05)) + 3
230 VTAB (A < 4) * 5 + (A > 3) * 12
240 CALL 768,13,3
250 GOTO 180
260 HOME : VTAB 20: PRINT "BYE!": END

```



```

1000 *SAVE S.ERASE (JEFF CREAMER)
1010 *-----*
1020 *                                     *
1030 *             ERASE ROUTINE             *
1040 *                                     *
1050 *             Jeff Creamer               *
1060 *                                     *
1070 *             CALL 768,(WIDTH),(DEPTH)   *
1080 *                                     *
1090 *-----*
1100 *             PAGE ZERO VARIABLES         *
1110 *-----*
0024- 1120 MON.CH      .EQ $24
0025- 1130 MON.CV      .EQ $25
1140 *-----*
1150 *             APPLESOFT ROUTINES USED     *
1160 *-----*
DEBE- 1170 AS.CHKCOM   .EQ $DEBE
E6F8- 1180 AS.GETBYT   .EQ $E6F8
E199- 1190 AS.IQERR    .EQ $E199
1200 *-----*
1210 *             MONITOR ROUTINES USED       *
1220 *-----*
FC22- 1230 MON.VTAB    .EQ $FC22
F94A- 1240 MON.PREL2    .EQ $F94A
1250 *-----*
1260 *             .OR $300
1270 *             .TF ERASE
1280 *-----*
1290 *             JEFF'S ERASE ROUTINE        *
1300 *-----*
0300- A5 25 1310 ERASE LDA MON.CV      GET VERTICAL COORD
0302- 48 1320 PHA                      SAVE ON STACK
0303- A5 24 1330 LDA MON.CH      AND HORIZ COORD
0305- 48 1340 PHA                      SAVE IT ON STACK, TOO
0306- 20 BE DE 1350 JSR AS.CHKCOM   COMMA?
0309- 20 F8 E6 1360 JSR AS.GETBYT   YES, GET WIDTH TO ERASE
030C- 8A 1370 TXA                      INTO ACC
030D- F0 3A 1380 BEQ .4             WIDTH MUST BE NON-ZERO
030F- 48 1390 PHA                      PUSH WIDTH ON STACK
0310- 20 BE DE 1400 JSR AS.CHKCOM   COMMA NEXT?
0313- 20 F8 E6 1410 JSR AS.GETBYT   YES, GET DEPTH
0316- 8A 1420 TXA                      AND TRANSFER TO ACC
0317- F0 30 1430 BEQ .4             DEPTH MUST BE NON-ZERO
0319- A8 1440 TAY                      DEPTH INTO Y REGISTER
031A- 68 1450 PLA                      WIDTH BACK OFF STACK
031B- 48 1460 PHA                      BUT KEEP IT THERE ALSO
031C- AA 1470 TAX                      AND INTO X-REG
031D- A5 25 1480 .1 LDA MON.CV      REMEMBER CV ON STACK
031F- 48 1490 PHA
0320- 20 4A F9 1500 JSR MON.PREL2    PRINT WIDTH # OF BLANKS
0323- 68 1510 PLA                      GET OLD CV OFF STACK
0324- 88 1520 DEY                      DECREMENT DEPTH
0325- F0 17 1530 BEQ .3             ZERO LINES LEFT?
0327- AA 1540 TAX                      OLD CV INTO X-REGISTER
0328- E8 1550 INX                      NEXT LINE
0329- E0 18 1560 CPX #24          OFF THE BOTTOM?
032B- 90 02 1570 BCC .2             NO, USE THIS ONE
032D- A2 00 1580 LDX #0            YES, WRAP BACK TO TOP
032F- 86 25 1590 .2 STX MON.CV
0331- 20 22 FC 1600 JSR MON.VTAB    ADJUST BASE ADDRESS
0334- 68 1610 PLA                      WIDTH OFF STACK
0335- AA 1620 TAX                      TO SET UP X AGAIN
0336- 68 1630 PLA                      HORIZ COORD OFF STACK
0337- 48 1640 PHA                      BUT MAINTAIN IT THERE ALSO
0338- 85 24 1650 STA MON.CH      AND RESTORE HCURSOR
033A- 8A 1660 TXA                      PUSH WIDTH BACK ON STACK
033B- 48 1670 PHA                      FOR NEXT TIME AROUND
033C- D0 DF 1680 .3 BNE .1          LOOP ALWAYS
033E- 68 1690 PLA                      POP WIDTH OFF
033F- 68 1700 PLA                      GET HORIZ COORDINATE
0340- 85 24 1710 STA MON.CH      AND RESTORE IT
0342- 68 1720 PLA                      GET VERTICAL COORDINATE
0343- 85 25 1730 STA MON.CV      RESTORE IT, TOO
0345- 20 22 FC 1740 JSR MON.VTAB    ADJUST BASE ADDRESS
0348- 60 1750 RTS                      DONE
0349- 4C 99 E1 1760 .4 JMP AS.IQERR  ILLEGAL QUANTITY ERROR

```

# With these tools you can really program!

Presenting a selection of the best programming tools for the entire Apple II family!

**S-C Macro Assembler** – Combined editor/assembler includes 29 commands and 20 directives; with macros, conditional assembly, global replace, edit, and more. Well-known for ease-of-use and powerful features. Thousands of users in over 30 countries and in every type of industry attest to its speed, dependability, and user-friendliness. Blends power, simplicity, and performance to provide the optimum capabilities to both beginning and professional programmers. With 100-page manual and reference card, \$80. (If you own an older version, call for an upgrade price.)

**Cross Assembler Modules** – Owners of the S-C Macro Assembler may add the ability to develop programs for other systems. We have modules for most of the popular chips, at very reasonable prices:

6800/01/02 .....	\$32.50
6805 .....	\$32.50
6809 .....	\$32.50
68000 .....	\$50.00
Z-80 .....	\$32.50
PDP-11 .....	\$50.00
8048 .....	\$32.50
8051 .....	\$32.50
8080/85 .....	\$32.50
1802/04/05 .....	\$32.50

All of the cross assemblers retain the full power of the S-C Macro Assembler. You can develop programs for burning into EPROMs, transfer through a data-link, or direct execution by some of the plug-in processor cards now on the market.

**Apple Assembly Line** – Monthly newsletter for assembly language programmers, beginner or advanced. Tutorial articles; advanced techniques; handy utility programs; commented listings of code in DOS and Apple ROMs; reviews of relevant new books, hardware, software; and more! \$18 per year (add \$3 for first class postage in USA, Canada, Mexico; add \$13 postage for other countries).

**ES-CAPE (Extended S-C Applesoft Program Editor)**  
– Full-function, interactive program editor for Applesoft.

For painless programming and accurate entry of programs from magazines: fast, easy changes; split-screen operation; global search and replace; automatic line numbers; keyboard macros; and more! \$60

**Flash!** – The Integer BASIC Compiler from Laumer Research. Transforms your programs into machine language so they run many times faster. Adds many new functions and features to the language, supporting hi-res graphics and more! Optional assembly source code creation for use with S-C Assembler. Requires Integer BASIC in RAM or ROM to edit source programs (not required for compiled programs). Source code of run-time package also available. Compiler, \$79; run-time source, \$39.

**Double Precision Floating Point for Applesoft** – For the scientist or engineer who is not satisfied with Applesoft's nine-digit precision. With this 2048-byte machine language extension to Applesoft, you can get 21-digit precision whenever you need it! Supports +, -, \*, /, INPUT, and PRINT. Subroutines coded in Applesoft for 21-digit precision math functions (sin, cos, sqrt, log, exp, etc.) are also included. And it's fast! Only \$50.

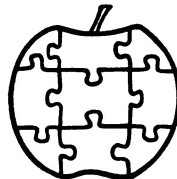
**DISASM** – Symbolic two-pass disassembler handles data tables, displaced object code, and includes a pre-defined symbol table of your choice. Produces ready-to-assemble source code with labels from machine code in ROM or RAM, formatted for S-C, Lisa, or ToolKit assemblers. \$30.

**S-C XREF** – Designed to work with the S-C Macro Assembler, produces a cross reference table of all labels used in your assembly source code. Only \$20, or \$50 with complete source code on disk.

All of our products are designed to be used. You can make back-up copies with any standard Apple disk copy program, and copies of the individual files can be moved to your working disks. We trust you.

S-C Software Corporation has frequently been commended for outstanding support: competent telephone help, continuing enhancements, and excellent upgrade policies. Call now and talk to Bob Sander-Cederlof or Bill Morgan. We want to help you program!

**S-C Software Corporation**  
2331 Gus Thomasson, Suite 125  
Dallas, Texas 75228  
(214) 324-2050



We take Mastercard, Visa, and American Express

Apple is a trademark of Apple Computer

The Amazing "quikLoader" Card.....Bob Sander-Cederlof

Jim Sather, author of "Understanding the Apple II", has designed and programmed a great new plug-in. It is basically a ROM card, but hold on to your hats!

The card has sockets for 8 EPROMs, and they can be any EPROM size from 2716 up through 27256. That means the card can hold a up to 256 kilobytes!

The card comes loaded already with three 2764 devices, programmed with licensed copies of DOS 3.3, FID, COPYA, the quikLoad operating system, and possibly more. I think Integer BASIC is on there too. With DOS on the card, you can leave it off your disks. You gain at least two tracks per disk this way.

The quikLoad operating system allows you to load any program from the card into RAM in a flash. If you have an EPROM programmer that can burn 2764s or larger, you can put favorites like the S-C Macro Assembler and our word processor permanently there. The programs don't even have to be modified, because they will be loaded into their normal RAM locations for execution.

You control the card by typing a control character along with RESET. For example, ctrl-C RESET catalogs a disk; ctrl-H RESET runs "HELLO"; others boot a disk or enter the monitor. Ctrl-Q RESET gives you a catalog of your quikLoader ROMs, in the form of a menu; a single keystroke then selects a program.

The board is compatible with Apple II, II Plus, and //e. In a II Plus with a 16K RAM card, you may need to perform a slight modification to the RAM card as explained in the documentation.

The boards are being manufactured by Southern California Research Group (SCRG), P. O. Box 2231, Goleta, CA 93118. Phone (805) 685-1931. Their price is \$179.50. You can order them from us if you like, at \$170 + shipping.

#### International Personal Robotics Conference

If you are among the many experimenting with little personal robots, such as Heathkit's HERO, you may be interested in attending the above named conference in Albuquerque next April 13-15. They are expecting around 4000 to show up from all over the world. You can meet such well known robotics experts as Joseph Engleberger, Nels Winless and others. It's a fair bet you'll find Jack Lewis of Micromation there. For more info, call Betty Bevers of IRPC at (303) 278-0662, or write to them at 1547 South Owens St. #46, Lakewood, Colorado 80226.

# Macro to Generate Quotient/Remainder Table for Hi-Res Work .....Bob Sander-Cederlof

A few months back an article in Byte magazine presented some fast hi-resolution plotting routines. One of the secrets to fast plotting is table lookup rather than computation of base addresses and offsets. The article included a 560 byte table for all the possible quotients and remainders you can get when dividing X by 7, where X is the horizontal coordinate (0 to 279).

The table of quotients and remainders makes it easy to get the byte position on a line (quotient) and the bit position in the byte (remainder) for a given dot X-coordinate.

Typing a 560 byte table into the computer is no fun, no matter how you do it. You might go into the monitor and type directly in hex, then later BSAVE the table. Or you might use an Applesoft program to build the table. I think the easiest way is to write a few short macros, and let the assembler do the work.

If you have Version 1.1 of the S-C Macro Assembler, the following code will do the trick. Version 1.0 cannot handle it, because the nesting level goes too deep. The listing it prints out gets quite long, due to all the macro expansion. Therefore I am just printing the source code here. The table it produces is also long, so I am just showing the beginning and end of it.

```

1000 *-----
1010 *   GENERATE QUOTIENT-REMAINDER
1020 *   TABLE FOR ALL POSSIBLE VALUES
1030 *   OF X/7, WHERE X=0...255
1040 *-----
1050       .MA DO.QS
1060 R       .SE 0
1070       >DO.RS
1080 Q       .SE Q+1
1090       .DO Q<40
1100       >DO.QS
1110       .FIN
1120       .EM
1130 *-----
1140       .MA DO.RS
1150       .DA #Q,#R
1160 R       .SE R+1
1170       .DO R<7
1180       >DO.RS
1190       .FIN
1200       .EM
1210 *-----
1220 Q       .SE 0
1230       >DO.QS
1240 *-----

```

0800-	00	00	00	01	00	02	00	03
0808-	00	04	00	05	00	06	10	00
0810-	01	01	01	02	01	03	01	04
:								
0A18-	26	02	26	03	26	04	26	05
0A20-	26	06	27	00	27	01	27	02
0A28-	27	03	27	04	27	05	27	06

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$12 postage for other countries. Back issues are available for \$1.50 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)